CS361 - Group 6 - HW6: Implementation 1
Sheng Bian | Michael Waldrop | Connor Shields | Linge Ge

**Software Access**

Our website can be accessed at http://web.engr.oregonstate.edu/~bians/361/index.html
This application can be directly accessed from desktop browser and mobile browser. No specicial instructions are required. This week we have implemented UI part of our project. It uses bootstrap framework to optimize user experience. Currently, the application has six UI pages, including homepage, guides list page, read guide page, view service page, sign in page and create new account page. Users can click buttons to browse different pages. The website has a top navigation bar on each web page so users can easily find the page they want to browse.

**Finished User Story Update**

Which pair(s) of teammates worked on that user story's tasks?

Initially, the pairs were assigned such that Connor and Sheng were to implement the create account page and subscribe to newsletter. Connor implemented the create account page and homepage, and Sheng compiled these contributions into the site with a more elaborate interface. Sheng also added a field on the homepage for the user to enter their email and subscribe to be on the site's newsletter. Michael and David worked on the service review functionality, which allows users to leave a star rating at the bottom of the service page as well as a section to add comments. Sheng also re-worked these contributions and implemented the rest of the site's interface.

What do the relevant unit tests do?

The unit test for using the create account page checks that the code verifies user input correctly. It passes input to the input fields automatically, once with a full submission of user data, then other tests with pieces missing (omitting one of each field at a time, to verify that the code catches it when these inputs aren't entered). It also sends in random inputs to password and password confirm, randomly capitalizing letters in the confirmation field to ensure that the system picks it up when there's a mismatch between passwords. These are also manually tested.

The login page could only be manually tested. We created an array of JavaScript objects with username and password fields, and stored these all on the login page. We tested with these inputs on the login page to ensure a user is only allowed to login if their username is in the array and if the password they enter matches the password associated with that username.

This is a website that is mostly concerned with information being presented in the proper place, so there aren't a lot of functions to test extensively. We tested the interface by navigating the site and making sure all the links worked.

What problems, if any, did you encounter?

The user rating was unresponsive at first, which was fixed easily by modifying the interface to allow a user to hover over a star to leave a rating.

There was also a problem with the create account page. It was expecting the user to enter an email in the first name field and to enter a first name in the email field. This was a problem, because the site is supposed to verify that an email address is valid. This was not being done on the account page, so there was no way for input to be validated properly. It was also a problem, because all users were required to add an "@" in their first name input. Sheng fixed this by swapping the validation code for the two fields. Along with the validation issue, there was no verification requiring that all the input fields be filled out. Sheng also fixed this by adding a condition requiring user input on every field before being able to successfully click "Submit".

How long did each task require?

We overestimated the time required for implementing the Rate and Review Service user story. Initially, we thought it would take 2 pair-programming hours, but it was easily completed in just one hour.

We also overestimated the time required for the create account and the newsletter subscription. Our implementation only took about 2 hours. Of course, this will take a little longer in a full implementation of the website and database, as we only implemented the front-end of our site so that it could be demonstrated for the customer.

What is the current status (implemented? tested?)

The user stories for creating an account and subscribing to a newsletter, "parents protecting their kids", and "rate and review service" were all implemented. Again, they are only completed as client-side code, so the full implementation will require a database connection that allows user information to be stored permanently. However, for our purposes, they are fully implemented and tested.

What is left to be completed?

We still need to implement the following user stories:

Viewing Guides in a Specific Category
View Top-Ranked Services
Manage Bookmarks

We already have the interface developed for some of these user stories, but we need to add code and test with a demo "database" to prove that we can render the web pages from a variety of data.

**UML Diagram/Spike Thoughts**

**User Story UML Diagram/Spike Evaluation**

- **Create account and subscribe to a newsletter UML sequence:** The UML diagram we designed was very helpful to create the frontend UI interface for creating an account. For example, we ensured that the user is able to access account creation and sign in from the homepage. After navigating to the account creation page, we followed the UML diagram closely to ensure that the user is able to input the correct data, with the correct data restrictions, and then submit that data. We know exactly where the data is coming from based on the diagram, and would be able to understand the need to setup an API to receive the data, process it, and send the appropriate authentication email. In summary, the UML diagram was very useful in that it allows us to stay on track on where we are and remembering the next steps during our implementation process.
- **Rate and Review Service:** The spike fairly accurately described the process for creating the star rating portion of the task, in that the actual implementation of the clickable rating system was accomplished in a combination of HTML, CSS, and JavaScript. Properly integrating this into the overall web page took much more effort and time, in accordance with the expectations listed in the spike.
- **Parents Protecting their Kids:** The UML diagram wasn't too helpful as a structure for us to design the whole process of viewing guides on the website. The data flow wasn't very useful, as the design is fairly straightforward, and something that we have been discussing since the beginning of this project that we didn't really need a UML diagram to help us understand the user story in order to build it better. However, it did help us indirectly. As we were reviewing the data flow of the user as they view guides, we realized they would benefit very much if they were able to see more information at the list of guides after they send in the request for all the guides. Being able to see more information at this stage of the diagram would reduce the number of times they would

have to click on individual guides. This led us to refactor the guides page so as to be able to include more preview information.

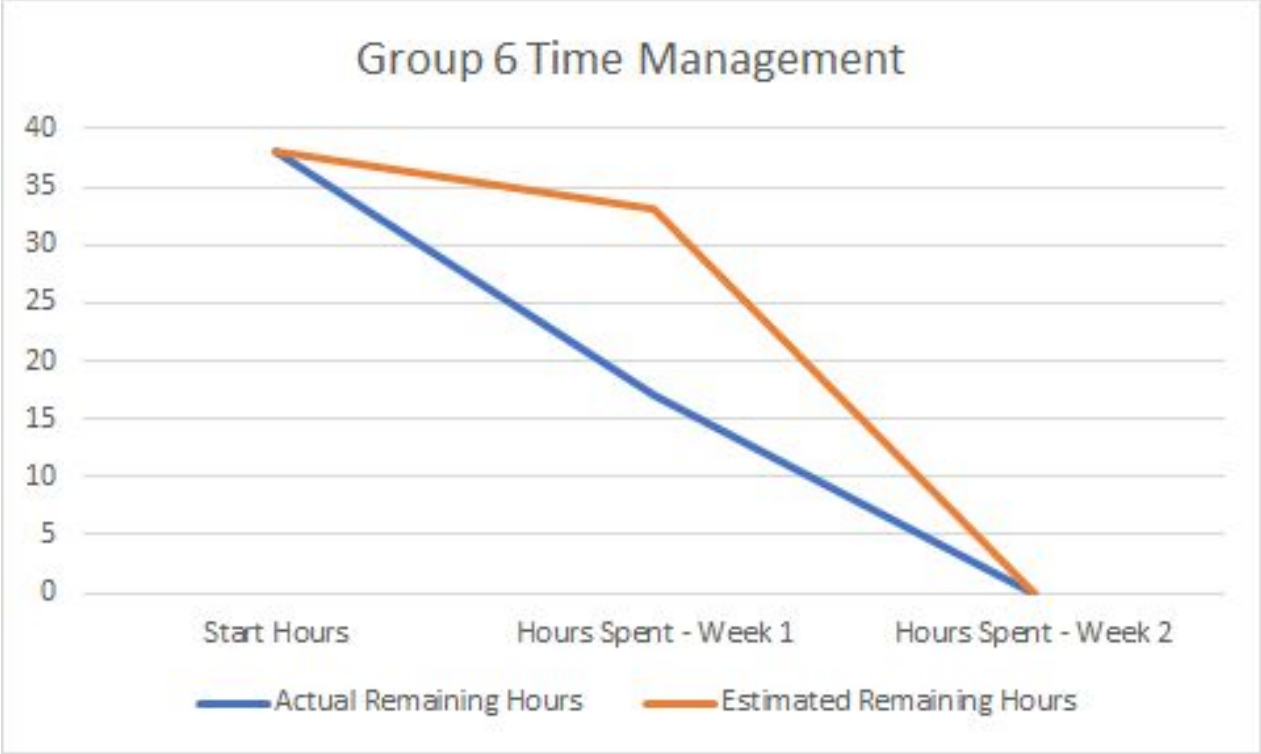
**UML Diagram**

We did not feel like we would need any more additional UML diagrams in order for us to implement the user stories. For the user stories in which we implemented already, we felt that the UML diagrams we designed have served their purposes very well. The user stories had very clear data flow and order of navigation in the website, which all of the existing UML diagrams we design already capture. Our existing UML diagrams are also very detailed, and having additional UML diagrams at this point would not only not be beneficial, but may lead to confusion as we have to consider the technical layout multiple UML diagrams present for the same case.

We didn't create a UML diagram for our rating system, but the spike that we did was more than sufficient in helping us understand what and how we should implement. Although the spike we did was not technically documented in detail, the experience of going through it allowed us to understand how to incorporate the review and rating system into our system, and how it would interact with the guides and services which the reviews would be targeting. At this point, we did have the optional to design and layout a UML diagram to summarize what we'll be doing before actually implementing it, and it may have been good in terms of documentation purposes. However, we implemented the feature just fine without it, and are very pleased with the how the feature developed and was integrated smoothly with the rest of the user stories and the architecture.

One UML sequence diagram that we think would have been helpful for our development process and progress is one where it maps out our entire website UI structure rather than follow a specific data flow path or user story. Our user stories and data flows go through and overlap with many UI pages and widgets on our website. This can lead to confusion as we design the UI and pages themselves, since we are doing the design based on a user-story implementation approach. For example, we designed the review and rating feature as well as a rating form and page for it. However, this is just standalone, and can be connected to the UI in several different ways for user to access and use it. At this point, the spike only mentions one way, but we recognize that the user can definitely access it in more than just one way that would be UX friendly. Having a UML diagram where we can map out different web pages and their interaction and connection would be helpful for us to see where on this map the user can access the system, and design it in such a way that is very clear for others on the team (ex: and not just randomly adding a link to it unbeknownst to others).

**Burndown Diagram**




**Refactoring Summary**

Originally we set up the development environment in one git repository that focused on using vanilla HTML/CSS/JavaScript to accomplish implementation of several of the user stories. However, later on we realized that building the web application with this process can be difficult, especially as we began implementing pipeline structures that require an efficient data flow from one page to another. Hence after building out several components for our user stories, we decided to do a complete overhaul.

We moved to a new environment where we rely on bootstrap and a modern business CSS template so that we are able to develop and focus on the business logic of the user features, instead of worrying about layout details and writing boilerplate code.

The following features were refactored:

- The homepage header was redesigned. Originally it had a title header with two buttons that allows a user to search through a list of guides and/or services, along with links to create an account. We have refactored to create a static header that will move the header from page to page regardless of where the user is on the website, so that a link is provided where they can always go back home if necessary. Instead of openly displaying “Search Guides” and “Search Services”, we have hidden it in a drop down menu that is available and accessible always.

- Newsletter subscription is now part of the homepage. Originally we had it set as a separate page, and provided a link from the homepage where the user can click on it, and then subscribe to it. Given that subscription is as easy as simply just providing an email, we recognized that there was no need to have a whole page dedicated to it. Furthermore, it would be more readily recognized and used if it was on the homepage at the very beginning. Hence we refactored the homepage so that it includes it.
- The homepage itself was refactored. Originally we kept it bare, in a Google search-esque format where a user must make a second click to see lists of services or guides, for the reason that it makes it very distinct for user (and in their stories), to make a decision on whether to find a specific service or read some guides on security topics. However, as a marketing tactic, it is useful to have some display of top services and guides (which was our original use case). We decided to go along with the use case design rather than the user story after consulting our customer, and now the homepage features some top ranked services and guides.
- The guides list was refactored to include more information. Originally it was a very simple layout with a search bar at the top, and a list of guides. This perfectly matches with our user stories, and would have been fine. However, we recognized that user stories can be enhanced if they were able to preview parts of each guide without having to fully commit to going to a different guide page. Hence, we refactored the list to not just be a list of guide names, but a list of previews of the guides themselves.
- Our create account page is refactored to include several conveniences such as the ability to sign in instead of creating an account, in case a user realizes they have an account. It also has some additional safeguards that did not exist before with the old system (due to the lack of ability to implement it on some forms), such as catching missing/blank information and the ability to remember user sign-in for their IP address.

**Customer Interaction**

The customer was sent a list of our proposed tasks, based on the earlier user stories we sent him. Each task was described in detail and assigned an estimated implementation time. The intention was for the customer to send us a response to let us know if this was an appropriate plan for implementing his vision, and to provide us with feedback if we were missing something important. The customer eventually did get back to us with a quick approval, stating that there was nothing else to add to feedback he gave us last week on the user stories and their task breakdowns. While we may have been hoping for a bit more explicit feedback on how things were going, it seems like the customer is more or less letting us lead in terms of the project development, and is only speaking up if anything seems really out of place. For example, the customer completely rejected a few of the user stories previously proposed because they did not fit his vision. For now, it looks like we're on the right track in developing this project.

Additionally, the customer was very pleased with our current progress and liked the website system we have designed and organized so far that will be the platform for the rest of the user stories. The customer has been very flexible with us, and besides the initial set of questions we asked while designing the user stories, have been supportive of all that he has seen us working on.

**Integration Tests**

Similar to what we found with unit testing, there was not much integration testing to do on our project, because at this stage, it only involves static web pages that authenticate user input and provides a friendly interface that is easy to navigate. The integration testing, then, mostly just consisted of making sure that all pages link together where they are supposed to. The user can click “sign in” from anywhere on the site’s domain, and that link works to correctly bring them to the login page. Our integration tests ensure that the URL for each link correctly goes to the intended page, and that the data from one page (if any) is successfully passed to the next page for usage. One big integration test was to ensure that the links in the header stay consistent from page to page, and there was nothing in the code that effectively changed then when a new page is loaded.

Along with this, they can create an account if they do not already have one by clicking on “create new account” from the sign in page. An integration test ensures that if a user successfully logs in and is remembered, they are at the very least consistently logged in throughout the duration of their stay on the website. The UI should reflect that state, and should not regress at any time to ask them to create a new account.

Similarly, a user always has the ability to click on “services” and “guides”, so they can easily access these pages to search for the content they are looking for. Once navigating to the services page, they can click on the service they want to read about, and leave a rating for that service. When navigating to the guides page, it allows them to see top guides, as well as categories for searching those guides. We do not have a database integrated with the site, but it is programmed to support data from a database if needed in the future. An integration test ensures that the rating and review system is correctly linked with services and guides so that clicking to rate on any service or guide page leads to a rating form that, once submitted, is submitted for that particular service or guide, and not any other.

Also on the guides page, several filtering and navigation options are available to the user. There is a search box, so they can search for a guide that matches their search (search functionality not implemented yet), and there is a marquee at the bottom of the page allowing them to navigate deeper into the list of services, which also is not implemented yet. This will require a

larger database, which will probably not be fully implemented. However, we will write the code to handle large amounts of data.

Overall, this week involved a lot of interface development, so the integration is limited, but we have everything in place to add more functionality to our page for next week's user stories.

**New Schedule For Next Week**

We did good work this week, and below serves as a summary for what we will tackle and focus on next week regarding the user stories we are implementing. The following are listed in order of relative importance and will be implemented top down:

- Viewing Guide In Specific Category (**Due Date:** 6/5)
    - **Tasks:** ~~Develop a homepage UI, and a clickable button leading to the guides page UI. From here, users can see a list of different guides in a populated list, with tabs at the top for category, date uploaded, rating, etc~~. Clicking on category will sort the guides by content type. Clicking on a guide name will list the title, author, date written, and a text block with information about the topic. At the bottom of the guide page, a small list will be populated with the most popular services for that content type.
    - **Est. Implementation Time:** 10 → **4** Pair-Programming Hours
    - **Programmer(s):** Sheng, Connor
- View top-ranked Services (**Due Date:** 6/5)
    - **Tasks:** Develop homepage UI and clickable services button. Clicking the services button will present a populated list of services sorted by rating. The top of the services list will have several clickable filter options, such as category for sorting services by specific technology types.
    - **Est. Implementation Time:** 9 Pair-Programming Hours
    - **Programmer(s):** Michael, Sheng
- Manage Bookmarks (**Due Date:** 6/5)
    - **Tasks:** Develop a user account homepage with a list of previously bookmarked guides and services based on the user's account identity. This page will have clickable delete buttons for the bookmarks, which will allow users to remove the bookmark form their user account database of bookmarked items. Develop a clickable bookmark button for each guide and service entry in the primary database that will link the viewed item to the user account identity, and copy/link the viewed item to the user's bookmark database.
    - **Est. Implementation Time:** 4 Pair-Programming Hours
    - **Programmer(s):** David, Connor
- Rate and Review Service (**Due Date:** ***Completed***)

    - **Nice-to-Have:** connecting our rate/review service form to each individual service page.
    - **Est. Implementation Time:** 2 → **1** Pair-Programming Hours
    - **Programmer(s):** Michael, David
- Parents Protecting Their Kids (**Due Date:** ***Completed***)
    - **Est. Implementation Time:** -
    - **Programmer(s):** Michael, David
- Create account and subscribe to newsletter for updates (**Due Date:** ***Completed***)
    - **Est. Implementation Time:** -
    - **Programmer(s):** Sheng, Connor

**Customer Contribution**

The customer responded to an email we sent him regarding our proposed plans for the week within 24 hours.

**Team Contribution Summary**

Sheng - Implemented bootstrap framework to all the UI pages. Refactored web pages to make it has the same style and mobile friendly. Deployed website on the flip server. Wrote access instructions.

Connor - Implemented code for create account page and homepage. Performed unit tests and wrote sections on unit testing and integration testing.

David - Implemented code for services list, services template and database. Linked homepage to services listed. Wrote summary on refactoring, work schedule for next week, customer contribution, UML/spike thoughts. Contacted customer to update on user stories and progress.

Michael - Wrote code for a star rating system with the ability to highlight 1-5 stars by clicking, a text entry box for writing a review, and clickable submit button. Created the burndown diagram for our time management for this week and our expected time management for next week. Wrote a summary of the customer's contributions for the week's work.